



ProDOS 8

#17: Recursive ProDOS Catalog Routine

Revised by: Dave Lyons, Keith Rollin, & Matt Deatherage

November 1989

Written by: Greg Seitz

December 1983

This Technical Note presents an assembly language example of a recursive directory reading routine which is AppleShare compatible.

Changes since November 1988: The routine now ignores the `file_count` field in a directory, and it properly increments `ThisBlock`. More discussion of AppleShare volumes is included.

This Note presents a routine in assembly language for recursively cataloging a ProDOS directory. If you apply this technique to the volume directory of a disk, it will display the name of every file stored on the disk. The routine displays the contents of a given directory (the volume directory in this case), displays the contents of each subdirectory as it is encountered.

`READ_BLOCK` is not used, since it does not work with AppleShare servers. `READ` is used instead, since it works for AppleShare volumes as well as local disks. Instead of using directory pointers to decide which block to read next, we simply read the directory and display filenames as we go, until we reach a subdirectory file. When we reach a subdirectory, the routine saves our place, plunges down one level of the tree structure, and catalogs the subdirectory. You repeat the process if you find a subdirectory at the current level. When you reach the EOF of any directory, the routine closes the current directory and pops back up one level, and when it reaches the EOF of the initial directory, the routine is finished.

This routine is generally compatible with AppleShare volumes, but it is impossible to guarantee a complete traversal of all the accessible files on an AppleShare volume: another user on the same volume can add or remove files or directories at any time. If entries are added or removed, some filenames may be displayed twice or missed completely. Be sure that your programs deal with this sort of situation adequately.

We assume that AppleShare is in short naming mode (as it is by default under ProDOS 8). If you enable long naming mode, then illegal characters in filenames will not be translated into question marks. In this case, the code would need to be modified to deal with non-ASCII characters. Also, the `ChopName` routine would need to be aware that a slash (/) character could be contained inside the name of a directory that had been added to the pathname. (As the code stands, such directories fail to open, but their names are still temporarily added to the pathname.)

When the catalog routine encounters an error, it displays a brief message and continues. It is important not to abort on an error, since AppleShare volumes generally contain files and folders with names that are inaccessible to ProDOS, as well as folders that are inaccessible to your program's user (error \$4E, access error).

The code example includes a simple test of the ReadDir routine, which is the actual recursive catalog routine. Note that the simple test relies upon the GETBUFR routine in BASIC.SYSTEM to allocate a buffer; therefore, as presented, the routine requires the presence of BASIC.SYSTEM. The actual ReadDir routine requires nothing outside of the ProDOS 8 MLI.

```
----- NEXT OBJECT FILE NAME IS CATALOG.0
0800:      0800      2      org      $800
0800:      3      *****
0800:      4      *
0800:      5      * Recursive ProDOS Catalog Routine
0800:      6      *
0800:      7      * by: Greg Seitz 12/83
0800:      8      *      Pete McDonald 1/86
0800:      9      *      Keith Rollin 7/88
0800:     10      *      Dave Lyons 11/89
0800:     11      *
0800:     12      * This program shows the latest "Apple Approved"
0800:     13      * method for reading a directory under ProDOS 8.
0800:     14      * READ_BLOCK is not used, since it is incompatible
0800:     15      * with AppleShare file servers.
0800:     16      *
0800:     17      * November 1989: The file_count field is no longer
0800:     18      * used (all references to ThisEntry were removed).
0800:     19      * This is because the file count can change on the fly
0800:     20      * on AppleShare volumes. (Note that the old code was
0800:     21      * accidentally decrementing the file count when it
0800:     22      * found an entry for a deleted file, so some files
0800:     23      * could be left off the end of the list.)
0800:     24      *
0800:     25      * Also, ThisBlock now gets incremented when a chunk
0800:     26      * of data is read from a directory. Previously, this
0800:     27      * routine could get stuck in an endless loop when
0800:     28      * a subdirectory was found outside the first block of
0800:     29      * its parent directory.
0800:     30      *
0800:     31      * Limitations: This routine cannot reach any
0800:     32      * subdirectory whose pathname is longer than 64
0800:     33      * characters, and it will not operate correctly if
0800:     34      * any subdirectory is more than 255 blocks long
0800:     35      * (because ThisBlock is only one byte).
0800:     36      *
0800:     37      *****
0800:     38      *
0800:     39      * Equates
0800:     40      *
0800:     41      * Zero page locations
0800:     42      *
0800:     0080     43      dirName      equ      $80          ; pointer to directory name
0800:     0082     44      entPtr      equ      $82          ; ptr to current entry
0800:     45      *
0800:     46      * ProDOS command numbers
0800:     47      *
0800:     BF00     48      MLI          equ      $BF00        ; MLI entry point
0800:     00C7     49      mliGetPfx    equ      $C7          ; GET_PREFIX
0800:     00C8     50      mliOpen      equ      $C8          ; Open a file command
0800:     00CA     51      mliRead      equ      $CA          ; Read a file command
0800:     00CC     52      mliClose     equ      $CC          ; Close a file command
```

```
0800:      00CE  53 mliSetMark equ  $CE          ; SET_MARK command
0800:      004C  54 EndOfFile equ  $4C          ; EndOfFile error
0800:              55 *
0800:              56 * BASIC.SYSTEM stuff
0800:              57 *
0800:      BEF5  58 GetBufr   equ   $BEF5          ; BASIC.SYSTEM get buffer routine
```

```

01 CATALOG                ProDOS Catalog Routine                14-OCT-89  16:20  PAGE 3

0800:                    59 *
0800:                    60 * Offsets into the directory
0800:                    61 *
0800:    0000    62 oType      equ    $0          ; offset to file type byte
0800:    0023    63 oEntLen    equ    $23         ; length of each dir. entry
0800:    0024    64 oEntBlk    equ    $24         ; entries in each block
0800:                    65 *
0800:                    66 * Monitor routines
0800:                    67 *
0800:    FDED    68 cout       equ    $FDED        ; output a character
0800:    FD8E    69 crout      equ    $FD8E        ; output a RETURN
0800:    FD8E    70 prbyte     equ    $FD8E        ; print byte in hex
0800:    00A0    71 space      equ    $A0          ; a space character
0800:                    72 *
0800:                    73 *****
0800:                    74 *
0800:    0800    75 Start      equ    *
0800:                    76 *
0800:                    77 * Simple routine to test the recursive ReadDir
0800:                    78 * routine. It gets an I/O buffer for ReadDir, gets
0800:                    79 * the current prefix, sets the depth of recursion
0800:                    80 * to zero, and calls ReadDir to process all of the
0800:                    81 * entries in the directory.
0800:                    82 *
0800:A9 04      83          lda    #4              ; get an I/O buffer
0802:20 F5 BE    84          jsr    GetBufr
0805:B0 17 081E  85          bcs    exit            ; didn't get it
0807:8D D7 09    86          sta    ioBuf+1
080A:                    87 *
080A:                    88 * Use the current prefix for the name of the
080A:                    89 * directory to display. Note that the string we
080A:                    90 * pass to ReadDir has to end with a "/", and that
080A:                    91 * the result of GET_PREFIX does.
080A:                    92 *
080A:20 00 BF    93          jsr    MLI
080D:C7          94          db     mliGetPfx
080E:E8 09      95          dw     GetPParms
0810:B0 0C 081E  96          bcs    exit
0812:                    97 *
0812:A9 00      98          lda    #0
0814:8D CE 09    99          sta    Depth
0817:                    100 *
0817:A9 EB      101          lda    #nameBuffer
0819:A2 0B      102          ldx    #<nameBuffer
081B:20 1F 08    103          jsr    ReadDir
081E:                    104 *
081E:    081E  105 exit      equ    *
081E:60          106          rts
081F:                    107 *
081F:                    108 *****
081F:                    109 *****
081F:                    110 *
081F:    081F  111 ReadDir    equ    *
081F:                    112 *
081F:                    113 * This is the actual recursive routine. It takes as
081F:                    114 * input a pointer to the directory name to read in
081F:                    115 * A,X (lo,hi), opens it, and starts to read the
081F:                    116 * entries. When it encounters a filename, it calls

```

```

01 CATALOG                ProDOS Catalog Routine                14-OCT-89  16:20  PAGE 4

081F:                    117 *  the routine "VisitFile". When it encounters a
081F:                    118 *  directory name, it calls "VisitDir".
081F:                    119 *
081F:                    120 *  The directory pathname string must end with a "/"
081F:                    121 *  character.
081F:                    122 *
081F:                    123 *****
081F:                    124 *
081F:85 80                125                sta  dirName          ; save a pointer to name
0821:86 81                126                stx  dirName+1
0823:                    127 *
0823:8D D4 09            128                sta  openName        ; set up OpenFile params
0826:8E D5 09            129                stx  openName+1
0829:                    130 *
0829:                0829 131 ReadDir1 equ  *                ; recursive entry point
0829:20 79 08            132                jsr  OpenDir          ; open the directory as a file
082C:B0 1F 084D          133                bcs  done
082E:                    134 *
082E:4C 48 08            135                jmp  nextEntry        ; jump to the end of the loop
0831:                    136 *
0831:                0831 137 loop    equ  *
0831:A0 00                138                ldy  #oType          ; get type of current entry
0833:B1 82                139                lda  (entPtr),y
0835:29 F0                140                and  #$F0          ; look at 4 high bits
0837:C9 00                141                cmp  #0          ; inactive entry?
0839:F0 0D 0848          142                beq  nextEntry        ; yes - bump to next one
083B:C9 D0                143                cmp  #$D0          ; is it a directory?
083D:F0 06 0845          144                beq  ItsADir          ; yes, so call VisitDir
083F:20 B3 08            145                jsr  VisitFile        ; no, it's a file
0842:4C 48 08            146                jmp  nextEntry
0845:                    147 *
0845:20 BA 08            148 ItsADir  jsr  VisitDir
0848:                0848 149 nextEntry equ  *
0848:20 77 09            150                jsr  GetNext          ; get pointer to next entry
084B:90 E4 0831          151                bcc  loop          ; Carry set means we're done
084D:                084D 152 done    equ  *                ; moved before PHA (11/89 DAL)
084D:48                    153                pha          ; save error code
084E:                    154 *
084E:20 00 BF            155                jsr  MLI          ; close the directory
0851:CC                    156                db  mliClose
0852:E1 09                157                dw  CloseParms
0854:                    158 *
0854:68                    159                pla          ;we're expecting EndOfFile error
0855:C9 4C                160                cmp  #EndOfFile
0857:F0 1F 0878          161                beq  hitDirEnd
0859:                    162 *
0859:                    163 *  We got an error other than EndOfFile--report the
0859:                    164 *  error clumsily ("ERR=$xx").
0859:                    165 *
0859:48                    166                pha
085A:A9 C5                167                lda  #'E'|$80
085C:20 ED FD            168                jsr  cout
085F:A9 D2                169                lda  #'R'|$80
0861:20 ED FD            170                jsr  cout
0864:20 ED FD            171                jsr  cout
0867:A9 BD                172                lda  #'='|$80
0869:20 ED FD            173                jsr  cout
086C:A9 A4                174                lda  #'$'|$80

```

```

01 CATALOG                ProDOS Catalog Routine                14-OCT-89  16:20  PAGE 5

086E:20 ED FD            175                jsr    cout
0871:68                  176                pla
0872:20 DA FD            177                jsr    prbyte
0875:20 8E FD            178                jsr    crout
0878:                    179 *
0878:                0878 180 hitDirEnd equ    *
0878:60                  181                rts
0879:                    182 *
0879:                    183 *****
0879:                    184 *
0879:                0879 185 OpenDir  equ    *
0879:                    186 *
0879:                    187 *  Opens the directory pointed to by OpenParms
0879:                    188 *  parameter block. This pointer should be init-
0879:                    189 *  ialized BEFORE this routine is called. If the
0879:                    190 *  file is successfully opened, the following
0879:                    191 *  variables are set:
0879:                    192 *
0879:                    193 *      xRefNum      ; all the refnums
0879:                    194 *      entryLen    ; size of directory entries
0879:                    195 *      entPtr      ; pointer to current entry
0879:                    196 *      ThisBEntry  ; entry number within this block
0879:                    197 *      ThisBlock   ; offset (in blocks) into dir.
0879:                    198 *
0879:20 00 BF            199                jsr    MLI                ; open dir as a file
087C:C8                  200                db      mliOpen
087D:D3 09              201                dw      OpenParms
087F:B0 31      08B2    202                bcs    OpenDone
0881:                    203 *
0881:AD D8 09           204                lda    oRefNum            ; copy the refnum return-
0884:8D DA 09           205                sta    rRefNum            ; ed by Open into the
0887:8D E2 09           206                sta    cRefNum            ; other param blocks.
088A:8D E4 09           207                sta    sRefNum
088D:                    208 *
088D:20 00 BF           209                jsr    MLI                ; read the first block
0890:CA                  210                db      mliRead
0891:D9 09              211                dw      ReadParms
0893:B0 1D      08B2    212                bcs    OpenDone
0895:                    213 *
0895:AD 0E 0A           214                lda    buffer+oEntLen ; init 'entryLen'
0898:8D D1 09           215                sta    entryLen
089B:                    216 *
089B:A9 EF              217                lda    #buffer+4        ; init ptr to first entry
089D:85 82              218                sta    entPtr
089F:A9 09              219                lda    #<buffer+4
08A1:85 83              220                sta    entPtr+1
08A3:                    221 *
08A3:AD 0F 0A           222                lda    buffer+oEntblk ; init these values based on
08A6:8D CF 09           223                sta    ThisBEntry    ; values in the dir header
08A9:8D D2 09           224                sta    entPerBlk
08AC:                    225 *
08AC:A9 00              226                lda    #0                ; init block offset into dir.
08AE:8D D0 09           227                sta    ThisBlock
08B1:                    228 *
08B1:18                 229                clc                ; say that open was OK
08B2:                    230 *
08B2:                08B2 231 OpenDone equ    *
08B2:60                  232                rts

```

```

01 CATALOG                ProDOS Catalog Routine                14-OCT-89  16:20 PAGE 6

08B3:                    233 *
08B3:                    234 *****
08B3:                    235 *
08B3:    08B3 236 VisitFile equ  *
08B3:                    237 *
08B3:                    238 * Do whatever is necessary when we encounter a
08B3:                    239 * file entry in the directory. In this case, we
08B3:                    240 * print the name of the file.
08B3:                    241 *
08B3:20 AC 09            242          jsr  PrintEntry
08B6:20 8E FD            243          jsr  crout
08B9:60                  244          rts
08BA:                    245 *
08BA:                    246 *****
08BA:                    247 *
08BA:    08BA 248 VisitDir equ  *
08BA:                    249 *
08BA:                    250 * Print the name of the subdirectory we are looking
08BA:                    251 * at, appending a "/" to it (to indicate that it's
08BA:                    252 * a directory), and then calling RecursDir to list
08BA:                    253 * everything in that directory.
08BA:                    254 *
08BA:20 AC 09            255          jsr  PrintEntry      ; print dir's name
08BD:A9 AF              256          lda  #'/'|$80      ; tack on / at end
08BF:20 ED FD            257          jsr  cout
08C2:20 8E FD            258          jsr  crout
08C5:                    259 *
08C5:20 C9 08            260          jsr  RecursDir      ; enumerate all entries in sub-
dir.
08C8:                    261 *
08C8:60                  262          rts
08C9:                    263 *
08C9:                    264 *****
08C9:                    265 *
08C9:    08C9 266 RecursDir equ  *
08C9:                    267 *
08C9:                    268 * This routine calls ReadDir recursively. It
08C9:                    269 *
08C9:                    270 * - increments the recursion depth counter,
08C9:                    271 * - saves certain variables onto the stack
08C9:                    272 * - closes the current directory
08C9:                    273 * - creates the name of the new directory
08C9:                    274 * - calls ReadDir (recursively)
08C9:                    275 * - restores the variables from the stack
08C9:                    276 * - restores directory name to original value
08C9:                    277 * - re-opens the old directory
08C9:                    278 * - moves to our last position within it
08C9:                    279 * - decrements the recursion depth counter
08C9:                    280 *
08C9:EE CE 09            281          inc  Depth      ; bump this for recursive call
08CC:                    282 *
08CC:                    283 * Save everything we can think of (the women,
08CC:                    284 * the children, the beer, etc.).
08CC:                    285 *
08CC:A5 83              286          lda  entPtr+1
08CE:48                  287          pha
08CF:A5 82              288          lda  entPtr
08D1:48                  289          pha
08D2:AD CF 09            290          lda  ThisBEntry

```

01 CATALOG	ProDOS Catalog Routine	14-OCT-89 16:20 PAGE 7
08D5:48	291 pha	
08D6:AD D0 09	292 lda ThisBlock	
08D9:48	293 pha	
08DA:AD D1 09	294 lda entryLen	
08DD:48	295 pha	
08DE:AD D2 09	296 lda entPerblk	
08E1:48	297 pha	
08E2:	298 *	
08E2:	299 * Close the current directory, as ReadDir will	
08E2:	300 * open files of its own, and we don't want to	
08E2:	301 * have a bunch of open files lying around.	
08E2:	302 *	
08E2:20 00 BF	303 jsr MLI	
08E5:CC	304 db mliClose	
08E6:E1 09	305 dw CloseParms	
08E8:	306 *	
08E8:20 2F 09	307 jsr ExtendName ; make new dir name	
08EB:	308 *	
08EB:20 29 08	309 jsr ReadDir1 ; enumerate the subdirectory	
08EE:	310 *	
08EE:20 65 09	311 jsr ChopName ; restore old directory name	
08F1:	312 *	
08F1:20 79 08	313 jsr OpenDir ; re-open it back up	
08F4:90 01 08F7	314 bcc reOpened	
08F6:	315 *	
08F6:	316 * Can't continue from this point--exit in	
08F6:	317 * whatever way is appropriate for your	
08F6:	318 * program.	
08F6:	319 *	
08F6:00	320 brk	
08F7:	321 *	
08F7: 08F7	322 reOpened equ *	
08F7:	323 *	
08F7:	324 * Restore everything that we saved before	
08F7:	325 *	
08F7:68	326 pla	
08F8:8D D2 09	327 sta entPerBlk	
08FB:68	328 pla	
08FC:8D D1 09	329 sta entryLen	
08FF:68	330 pla	
0900:8D D0 09	331 sta ThisBlock	
0903:68	332 pla	
0904:8D CF 09	333 sta ThisBEntry	
0907:68	334 pla	
0908:85 82	335 sta entPtr	
090A:68	336 pla	
090B:85 83	337 sta entPtr+1	
090D:	338 *	
090D:A9 00	339 lda #0	
090F:8D E5 09	340 sta Mark	
0912:8D E7 09	341 sta Mark+2	
0915:AD D0 09	342 lda ThisBlock ; reset last position in dir	
0918:0A	343 asl a ; = to block # times 512	
0919:8D E6 09	344 sta Mark+1	
091C:2E E7 09	345 rol Mark+2	
091F:	346 *	
091F:20 00 BF	347 jsr MLI ; reset the file marker	
0922:CE	348 db mliSetMark	

```

01 CATALOG          ProDOS Catalog Routine          14-OCT-89  16:20 PAGE 8

0923:E3 09          349          dw      SetMParms
0925:              350 *
0925:20 00 BF      351          jsr      MLI          ; now read in the block we
0928:CA              352          db      mliRead      ; were on last.
0929:D9 09          353          dw      ReadParms
092B:              354 *
092B:CE CE 09      355          dec      Depth
092E:60              356          rts
092F:              357 *
092F:              358 *****
092F:              359 *
092F:          092F 360 ExtendName equ *
092F:              361 *
092F:              362 * Append the name in the current directory entry
092F:              363 * to the name in the directory name buffer. This
092F:              364 * will allow us to descend another level into the
092F:              365 * disk hierarchy when we call ReadDir.
092F:              366 *
092F:A0 00          367          ldy      #0          ; get length of string to copy
0931:B1 82          368          lda      (entPtr),y
0933:29 0F          369          and      #$0F
0935:8D 62 09      370          sta      extCnt      ; save the length here
0938:8C 63 09      371          sty      srcPtr      ; init src ptr to zero
093B:              372 *
093B:A0 00          373          ldy      #0          ; init dest ptr to end of
093D:B1 80          374          lda      (dirName),y ; the current directory name
093F:8D 64 09      375          sta      destPtr
0942:              376 *
0942:          0942 377 extloop equ *
0942:EE 63 09      378          inc      srcPtr      ; bump to next char to read
0945:EE 64 09      379          inc      destPtr     ; bump to next empty location
0948:AC 63 09      380          ldy      srcPtr      ; get char of sub-dir name
094B:B1 82          381          lda      (entPtr),y
094D:AC 64 09      382          ldy      destPtr     ; tack on to end of cur. dir.
0950:91 80          383          sta      (dirName),y
0952:CE 62 09      384          dec      extCnt      ; done all chars?
0955:D0 EB 0942    385          bne      extloop     ; no - so do more
0957:              386 *
0957:C8              387          iny
0958:A9 2F          388          lda      #'/'          ; tack "/" on to the end
095A:91 80          389          sta      (dirName),y
095C:              390 *
095C:98              391          tya          ; fix length of filename to open
095D:A0 00          392          ldy      #0
095F:91 80          393          sta      (dirName),y
0961:              394 *
0961:60              395          rts
0962:              396 *
0962:          0001 397 extCnt ds 1
0963:          0001 398 srcPtr ds 1
0964:          0001 399 destPtr ds 1
0965:              400 *
0965:              401 *
0965:              402 *****
0965:              403 *
0965:          0965 404 ChopName equ *
0965:              405 *
0965:              406 * Scans the current directory name, and chops

```

```

01 CATALOG                ProDOS Catalog Routine                14-OCT-89  16:20  PAGE  9

0965:                    407 * off characters until it gets to a /.
0965:                    408 *
0965:A0 00                409             ldy    #0                ; get len of current dir.
0967:B1 80                410             lda    (dirName),y
0969:A8                    411             tay
096A:                    096A 412 ChopLoop equ    *
096A:88                    413             dey                ; bump to previous char
096B:B1 80                414             lda    (dirName),y
096D:C9 2F                415             cmp    #'/'
096F:D0 F9                096A 416             bne    ChopLoop
0971:98                    417             tya
0972:A0 00                418             ldy    #0
0974:91 80                419             sta    (dirName),y
0976:60                    420             rts
0977:                    421 *
0977:                    422 *****
0977:                    423 *
0977:                    0977 424 GetNext  equ    *
0977:                    425 *
0977:                    426 * This routine is responsible for making a pointer
0977:                    427 * to the next entry in the directory. If there are
0977:                    428 * still entries to be processed in this block, then
0977:                    429 * we simply bump the pointer by the size of the
0977:                    430 * directory entry. If we have finished with this
0977:                    431 * block, then we read in the next block, point to
0977:                    432 * the first entry, and increment our block counter.
0977:                    433 *
0977:CE CF 09            434             dec    ThisBEntry    ; dec count for this block
097A:F0 10            098C 435             beq    ReadNext    ; done w/this block, get next one
097C:                    436 *
097C:18                437             clc                ; else bump up index
097D:A5 82            438             lda    entPtr
097F:6D D1 09          439             adc    entryLen
0982:85 82            440             sta    entPtr
0984:A5 83            441             lda    entPtr+1
0986:69 00            442             adc    #0
0988:85 83            443             sta    entPtr+1
098A:18                444             clc                ; say that the buffer's good
098B:60                445             rts
098C:                    446 *
098C:                    098C 447 ReadNext equ    *
098C:20 00 BF          448             jsr    MLI                ; get the next block
098F:CA                449             db    mliRead
0990:D9 09                450             dw    ReadParms
0992:B0 16            09AA 451             bcs    DirDone
0994:                    452 *
0994:EE D0 09          453             inc    ThisBlock
0997:                    454 *
0997:A9 EF            455             lda    #buffer+4    ; set entry pointer to beginning
0999:85 82            456             sta    entPtr    ; of first entry in block
099B:A9 09            457             lda    #<buffer+4
099D:85 83            458             sta    entPtr+1
099F:                    459 *
099F:AD D2 09          460             lda    entPerBlk    ; re-init 'entries in this block'
09A2:8D CF 09          461             sta    ThisBEntry
09A5:CE CF 09          462             dec    ThisBEntry
09A8:18                463             clc                ; return 'No error'
09A9:60                464             rts

```

```

01 CATALOG                ProDOS Catalog Routine                14-OCT-89  16:20  PAGE 10

09AA:                    465 *
09AA:                    466 DirDone    equ    *
09AA:38                  467            sec                ; return 'an error occurred' (error
in A)
09AB:60                  468            rts
09AC:                    469 *
09AC:                    470 *****
09AC:                    471 *
09AC:                    09AC 472 PrintEntry equ    *
09AC:                    473 *
09AC:                    474 * Using the pointer to the current entry, this
09AC:                    475 * routine prints the entry name. It also pays
09AC:                    476 * attention to the recursion depth, and indents
09AC:                    477 * by 2 spaces for every level.
09AC:                    478 *
09AC:AD CE 09           479            lda    Depth                ; indent two blanks for each
level
09AF:0A                  480            asl    a                ; of directory nesting
09B0:AA                  481            tax
09B1:F0 08              09BB 482            beq    spcDone
09B3:A9 A0              483            lda    #space
09B5:20 ED FD           484            jsr    cout
09B8:CA                  485            dex
09B9:D0 F8              09B3 486            bne    spcloop
09BB:                    09BB 487            spcDone    equ    *
09BB:                    488 *
09BB:A0 00              489            ldy    #0                ; get byte that has the length
byte
09BD:B1 82              490            lda    (entPtr),y
09BF:29 0F              491            and    #$0F                ; get just the length
09C1:AA                  492            tax
09C2:                    09C2 493 PrntLoop    equ    *
09C2:C8                  494            iny                ; bump to the next char.
09C3:B1 82              495            lda    (entPtr),y        ; get next char
09C5:09 80              496            ora    #$80                ; COUNT likes high bit set
09C7:20 ED FD           497            jsr    cout                ; print it
09CA:CA                  498            dex                ; printed all chars?
09CB:D0 F5              09C2 499            bne    PrntLoop        ; no - keep going
09CD:60                  500            rts
09CE:                    501 *
09CE:                    502 *****
09CE:                    503 *
09CE:                    504 * Some global variables
09CE:                    505 *
09CE:                    0001 506 Depth      ds    1                ; amount of recursion
09CF:                    0001 507 ThisBEntry ds    1                ; entry in this block
09D0:                    0001 508 ThisBlock ds    1                ; block with dir
09D1:                    0001 509 entryLen  ds    1                ; length of each directory entry
09D2:                    0001 510 entPerBlk ds    1                ; entries per block
09D3:                    511 *
09D3:                    512 *****
09D3:                    513 *
09D3:                    514 * ProDOS command parameter blocks
09D3:                    515 *
09D3:                    09D3 516 OpenParms equ    *
09D3:03                  517            db    3                ; number of parms
09D4:                    0002 518 OpenName  ds    2                ; pointer to filename
09D6:00 00              519            ioBuf   dw    $0000        ; I/O buffer
09D8:                    0001 520 oRefNum   ds    1                ; returned refnum
09D9:                    521 *
09D9:                    09D9 522 ReadParms equ    *

```

```

01 CATALOG                ProDOS Catalog Routine                14-OCT-89  16:20 PAGE 11

09D9:04                   523                db      4                ; number of parms
09DA:                   0001 524 rRefNum      ds      1                ; refnum from Open
09DB:EB 09               525                dw      buffer           ; pointer to buffer
09DD:00 02               526 reqAmt        dw      512              ; amount to read
09DF:                   0002 527 retAmt      ds      2                ; amount actually read
09E1:                   528 *
09E1:                   09E1 529 CloseParms equ *
09E1:01                  530                db      1                ; number of parms
09E2:                   0001 531 cRefNum      ds      1                ; refnum from Open
09E3:                   532 *
09E3:                   09E3 533 SetMParms equ *
09E3:02                  534                db      2                ; number of parms
09E4:                   0001 535 sRefNum      ds      1                ; refnum from Open
09E5:                   0003 536 Mark        ds      3                ; file position
09E8:                   537 *
09E8:                   09E8 538 GetPParms equ *
09E8:01                  539                db      1                ; number of parms
09E9:EB 0B               540                dw      nameBuffer       ; pointer to buffer
09EB:                   541 *
09EB:                   0200 542 buffer      ds      512             ; enough for whole block
09EB:                   543 *
09EB:                   0040 544 nameBuffer ds      64                ; space for directory name

```

```

01 SYMBOL TABLE          SORTED BY SYMBOL                      14-OCT-89  16:20 PAGE 12

09EB BUFFER               096A CHOPLOOP               0965 CHOPNAME               09E1 CLOSEPARMS
FDED COUT                 09E2 CREFNUM               FD8E CROUT                 09CE DEPTH
0964 DESTPTR              09AA DIRDONE                80 DIRNAME                084D DONE
  4C ENDOFFILE            09D2 ENTPERBLK              82 ENTPTR                 09D1 ENTRYLEN
081E EXIT                 0962 EXTCNT               092F EXTENDNAME           0942 EXTLOOP
BEF5 GETBUFR              0977 GETNEXT               09E8 GETPPARMS           0878 HITDIREND
09D6 IOBUF                0845 ITSADIR               0831 LOOP                 09E5 MARK
  CC MLICLOSE             C7 MLIGETPFIX              C8 MLIOPEN                BF00 MLI
  CA MLIREAD              CE MLISETMARK            0BEB NAMEBUFFER           0848 NEXTENTRY
  24 OENTBLK              23 OENTLEN              0879 OPENDIR              08B2 OPENDONE
09D4 OPENNAME            09D3 OPENPARMS           09D8 OREFNUM              00 OTYPE
FDDA PRBYTE              09AC PRINTENTRY          09C2 PRNTLOOP            0829 READDIR1
081F READDIR             098C READNEXT            09D9 READPARMS           08C9 RECURSDIR
08F7 REOPENED            ?09DD REQAMT              ?09DF RETAMT              09DA RREFNUM
09E3 SETMPARMS           A0 SPACE                 09BB SPCDONE              09B3 SPCLOOP
0963 SRCPTR              09E4 SREFNUM              ?0800 START               09CF THISBENTRY
09D0 THISBLOCK           08BA VISITDIR            08B3 VISITFILE

** SUCCESSFUL ASSEMBLY := NO ERRORS
** ASSEMBLER CREATED ON 15-JAN-84 21:28
** TOTAL LINES ASSEMBLED 544
** FREE SPACE PAGE COUNT 81

```

Further Reference

- *ProDOS 8 Technical Reference Manual*
- *AppleShare Programmer's Guide to the Apple IIGS*